1995

Brokered Collaborative Infrastructure for CSCW

Chandrajit Bajaj

Peinan Zhang

Alok R. Chaturvedi
Purdue University, alok@cs.purdue.edu

Report Number:
95-001

BROKERED COLLABORATIVE INFRASTRUCTURE FOR CSCW

Chandrajit Bajaj
Peinan Zhang
Alok Chaturvedi

CSD-TR-95-001
January 1995

Brokered Collaborative Infrastructure for CSCW[*]

Chandrajit Bajaj Peinan Zhang
Department of Computer Sciences

Alok Chaturvedi
Krannert School of Management

Purdue University
West Lafayette, IN 47907

Email: {bajaj@cs,pnz@cs,alokrc@mgmt}.purdue.edu
Phone: (317)494-6531 Fax: (317)494-0739

Abstract

We demonstrate the advantages of a distributed collaborative system for CSCW, and highlight the requirements of brokered support for such a system. We also demonstrate how we have augmented the infrastructure of a prototype CSCW environment called SHASTRA to accommodate brokered collaboration. Several applications and possible scenarios of CSCW with brokered cooperative system are also presented. We describe how brokers can be used to exploit plurality and commonality of tasks in a cooperative setting, improving performance for the entire system.

Keywords: CSCW Infrastructure; Groupware; Task Brokering; Scheduling; Load Balancing; Cooperative Problem Solving;

[*]Supported in part by NSF grant CCR 92-22467, AFOSR grant F49620-93-10138, F49620-94-1-0080, and ONR grant N000014-94-1-0370

1 Introduction

Successful implementations of distributed CSCW application systems have been extremely limited, although there are many well developed single user distributed applications [8][15][12][17]. The primary reasons are that distributed CSCW systems are complex and difficult to build.

The difficulties in developing CSCW systems or in porting single user distributed applications to multi-user applications arise from:

1. Special distributed features caused by heterogeneous environments, and imbalance of user's motivations and activities.

2. The contradiction between transparency in distributed systems and awareness in CSCW systems.

3. Absence of general models and enabling infrastructure for collaboration within a group as well as among groups.

The motivation of this paper is to provide a paradigm for the development of distributed collaborative systems. We hope our system can not only make collaborative applications easy to build, but also can make single user applications easy to port to multiuser collaborative applications. To achieve our goal, we define a new model for collaboration, which can support collaboration not only within a group, but also among groups, and which can provide great flexibility, sharing ability and extensibility. Based on this model, we build an infrastructure, which relies on the concept of an Object Request Broker. Our brokered collaborative infrastructure provides an efficient support for managing a hierarchy of sessions, allows a flexible connection and communication in a distributed environment, and makes CSCW systems easy to integrate and extend. Finally, we develop distributed collaborative applications upon the infrastructure to demonstrate its feasibility.

The rest of this paper is organized as follows: section 2 describes related work; section 3 briefly introduces our environment; section 4 presents our brokered collaboration model and special CSCW design issues; section 5 describes our brokered collaborative infrastructure; section 6 shows some applications as our initial results; at last, section 7 addresses the features of our system and future direction.

2 Related Works

Groupware focuses on using the computer to facilitate human interaction for problem solving. Ellis *et al* present an overview of the field in [7]. The Rendezvous system proposes a powerful architecture for multi-user applications and provides high level support for creating groupware [14]. Language based approaches to generating multi-user applications are described in [10]. GroupKit presents a mechanism for creation of realtime work surfaces which are essentially shared visual environments [16]. Weasel is another system for implementing multi-user applications [9]. Networked collocation facilities have also received considerable attention *e.g.* MMConf [6], Rapport [1], etc. They provide useful conference management facilities, and support content-independent shared view-spaces.

All these system are built either by a centralized model, which may lose flexibility; or by a replicated model, which may have limitations in shared input or computation. So most of them are fail to satisfy flexibility and sharing requirement at the same time. Another common problem with these systems is that they only consider the collaborate transition within a group but not among groups.

Object Request Broker is one of useful concept in current distributed systems. Object Management Group (OMG), an industrial consortium, proposed the Common Object Request Broker Architecture (CORBA) [13], which was adopted from a joint proposal of the constituent companies (DEC, Hewlett-Packard , HyperDesk, NCR, Object Design, and SunSoft). The document defines a framework for different Object Request Broker (ORB) implementations to provide common services and interfaces to support portable clients and implementation objects. In the design arena, brokers can be used to access servers for analyses, simulations, animations, and other special purpose computation not locally available in an application. They also can be used to conduct database and file system searches in information systems. In general, brokers can provide clients a full transparent access to services, and make distributed systems easy to integrate and extend.

3 Highlights of Shastra

Shastra [1] is an extensible, distributed and collaborative geometric design and scientific manipulation environment. The CSCW infrastructure of the Shastra system facilitates creation of collaborative multimedia applications [3]. We adopt an abstract application architecture that enables inter-application communication and cooperation. The Shastra system architecture is described in detail in [3]. Example collaborative multimedia applications are described in [2]. Shastra consists of a static and a dynamic component. The static compo-

[1]Shastra is the Sanskrit word for Science

nent, the Shastra layer, is a CSCW infrastructure for building scientific CSCW applications. It defines an architectural paradigm that specifies guidelines on how to construct applications which are amenable to interoperation. Its connection and distribution substrate facilitates inter-application cooperation and distributed problem solving for concurrent engineering. Its communication substrate supports transport of multimedia information. The collaboration substrate supports building collaboration-aware synchronous multi-user applications by providing session management and access regulation facilities. In addition to the distribution, communication and collaboration framework, Shastra provides a powerful numeric, symbolic and graphics substrate. It enables rapid prototyping and development of collaborative software tools for the creation, manipulation and visualization of multi-dimensional geometric data.

Since Shastra is an extensively flexible system, it is easy to design a new model which can extend the collaboration functions not only within a group but also among groups to meet diverse requirements of practical applications. Considering the operating environment is distributed and heterogeneous, we need to find well developed distributed techniques to provide a flexible support.

4 Model

The main consideration of our model design is the flexibility. We define a policy-free support and control mechanism that lets an application or a user make policy decision to satisfy the different requirements of CSCW applications. We call this mechanism brokered collaborative model.

A broker is a customizable agent that functions as an intermediary between clients and servers. It allows applications to locate required servers and connect and communicate with them. A broker's list of tasks includes:

- Directory service.
- Locate suitable server in a directory to obtain best "price" and performance.
- Co-ordinate tasks between servers and balance loads.
- Notify other brokers if s new service is added or removed from the system.

The brokered collaborative model includes two layers, collaboration layer and distribution layer.

4.1 Collaboration Layer

The collaboration layer consists of a set of tools, which can be identical or different, and a central session manager. A session is a unit of collaborative activities. The central manager simplifies the synchronization control among tools.

4.1.1 Single Session

A single session in the Shastra model consists of a set of replicated tools at each site. To begin a collaborative activity, a session is started and a session manager is created. The session manager is responsible for setting up the connection to shared context inside a session and maintaining a view of the shared state. When a user joins a session, the session manager is responsible for creating shared context to add to session context; while it will tear them down when a user leaves the session. When a user modifies the session context, this modification will be reflected to all its members, shared by its members. Therefore session supports collaboration awareness.

4.1.2 A Hierarchy of Sessions

A hierarchical structure of sessions is defined to support collaboration among groups. Each session in the hierarchy can be a single session or a group session. A single session contains identical members, while a group session contains different members. When a session want to collaborate with other sessions, a group session is started, and a group session manager is created. As in a single session, the group session manager is responsible for handling collaboration control among sessions, and maintaining the group session context, which is a shared context among those sessions.

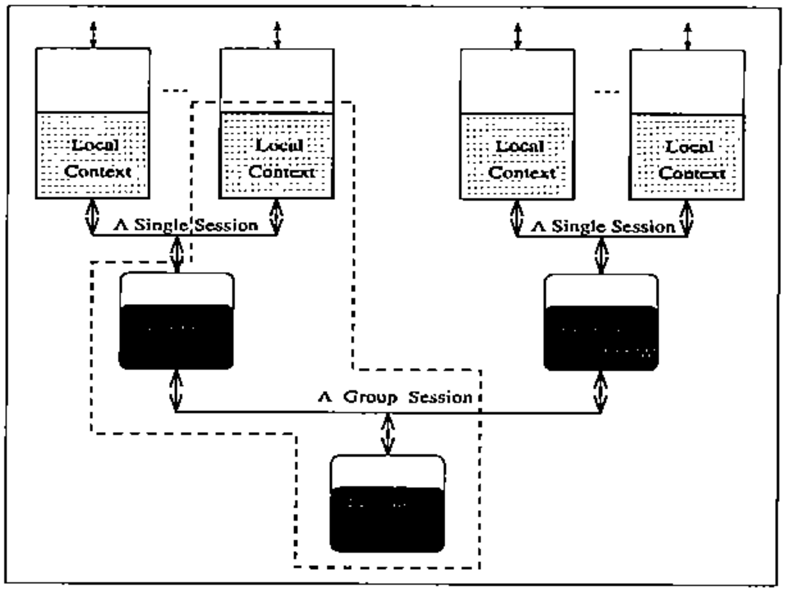


FIGURE 1: A hierarchy od sessions for collaboration model

As Figure 1 shows, the local context, the single session context and the group session context, make the total view of the application on a site. Since the group session context is the view of a group shared state, any changes of a group session context are visible to all its members. So the hierarchy of sessions support collaborative session awareness. By extending this structure, we can provide a hierarchy of sharing among users.

4.1.3 Main Design Issues

Among all the design issues involved in building a collaborative system, there are several distinguishing issues which make our model unobtainable by simple extend of traditional models.

- **Session control:** Because we support a hierarchy of sessions, compared to the traditional models, there are many more sessions in our system. Thus naming of sessions and the overhead of session managers will become problems.

- **Access control:** In traditional models, user's access right is described by either capability or permission. In our model, we have two kinds of sessions, single session with identical members and group session with different members. Therefore, simply using either one will become insufficient.

- **Floor control:** In traditional models, sessions are independent of others, so granting of the floor can be determined by the session manager itself; while in our model, there may be relationships among sessions in the hierarchy, so granting of the floor in a session may depend on other sessions, and likewise relinquishing the floor may also affect other sessions.

We will address these different requirements in our implementation.

4.2 Distribution Layer

Since we are considering a practical distributed collaborative environment, there are communications in distributed computation and in multiusers collaboration. This layer defines a mechanism to support a flexible connection and communication in the whole system. A dynamic scheduling method is designed to maintain system load balance in a distributed setting, so that a better performance can be achieved.

Another important issue defined in this layer is common task sharing. Since in a CSCW application, multi-users always work in the same working space, common task sharing will become important for optimization. By this mechanism, our model can avoid the disadvantage of replicated CSCW systems and satisfy both the sharing ability and the flexibility at the same time.

5 Brokered Collaborative Infrastructure

We introduce the concept of Object Request Broker [13] to implement our brokered collaborative infrastructure. In the following sections, we will describe the two most important substrates of our model, collaboration substrate and distribution substrate.

5.1 Collaboration Substrate

This fulfills the need of collaboration control, and provides a mechanism to implement multi-user interaction within a session and among sessions.

5.1.1 Session Control

This substrate provides mechanism to implement the management of the hierarchy of sessions, which include initiating a session, inviting users into a session, requesting to join or leave a session, and terminating a session. The key issue here is how to implement our session manager, since there are many more session managers in a hierarchy of sessions compared to a single session model. There are two extremes in implementation. In one extreme, each session manager is implemented by an independent session management server. While gets a distributed performance, it also suffers a huge overhead of session managers. In the other extreme, using a central server to manage all sessions reduces overhead, but the central server can become a bottleneck to cause a bad performance. To solve this problem, we introduced the concept of Object Request Broker into our system. Broker works as the substitute of a session manager to each session. A set of session management servers are connected to a broker to do the job. So for a session, broker works as if it were the session manager. Actually, broker just routes the request to one of session manager servers to handle the request. See Figure 2.

For certain applications, if the load on the session manager is light, then we can simply use one server to manage all sessions and achieve the lowest overhead of centralized management. If there are sessions with heavy tasks, then we can use more than one server for the session management to avoid the bottleneck. Therefore, by dynamically changing the number of session management servers, this implementation makes it possible to transparently move from centralized approach to a full distributed approach, so we can get distributed performance with a centralized overhead. Because of broker, this change cannot affect the

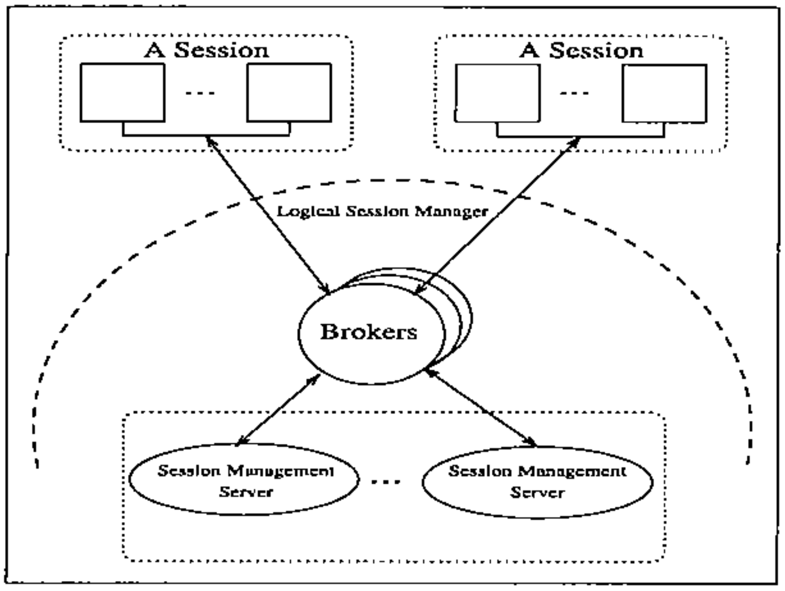


FIGURE 2: Brokered session management

sessions.

5.1.2 Access Control

Since the hierarchy consists of two kind of sessions, we use a hybrid access control scheme in our system. For a single session, which contains identical members, we describe user's access right in a capability list, which will allow users to have independent right; while in a group session, which contains different members, it is insufficient to use the same capabilities to describe users ability, so we use a permission to define user's access right to simplify the implementation.

5.1.3 Floor Control

Floor dependency is the major point in the floor control. A request propagation mechanism is designed. Here we use a single floor situation to describe how our system is implemented. In the hierarchy of sessions, an object can be shared not only within a session, but also among sessions. Suppose an object is only shared in a session A and if a user in session A requests the floor, the session manager of session A can handle the request immediately. But if the object is also shared with other sessions, that means session A is joining another group session G, the session manager of session A needs to get the floor in the session G first, only after this, it can handle that request. So we can see the request of a floor by a member in a session may cause the request of a floor by a session manager in its upper (group) session.

5.2 Distribution Substrate

This fulfills the need of distribution control, and provides a mechanism to implement connection and communication. Brokers introduced in this substrate play an important role in the implementation.

A broker works in a server-based model as classified by CORBA[13], that is clients and servers can communicate with a broker, and the broker's job is to route requests from clients to servers, and pass the results back if needed. Multiple brokers are introduced in our system, see Figure 3.

To reduce system bottleneck and risk of failure, servers in our system are divided into groups and one broker is only responsible for one group, and several brokers can work for the same server group. All broker in the system can communicate with each other as a client-server setting.

Several functionalities are implemented in the broker.

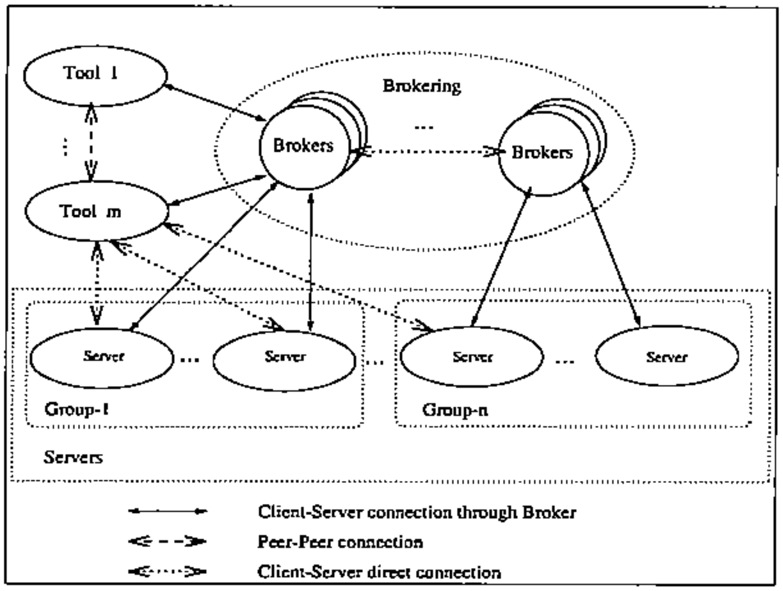


FIGURE 3: Brokered connection and communication

5.2.1 Flexible Connection

Broker can provide a full transparent connection and communication between clients and servers as in a common object request broker system. Furthermore, our system can support a more flexible control. Brokers provide several kinds of system information, such as what hosts or services are in the system, are they busy or idle, what are their costs or speed etc. Based on these information, users can select the connection according to their time and cost consideration. At last, broker can be by-passed, that is, clients can communicate to servers directly, this will be much cheaper whenever the message passing is heavier than task computing.

Here are the basic requests and responses between clients and brokers :

- HostInfo_Request/HostInfo_Notify: request/response current system host information.
- ServiceInfo_Request/ServiceInfo_Notify: request/response current system service in-

formation.

- Schedul_Request/Schedul_Notify: request/response several possible schedules for a certain number of request to a certain kind of services. The total response time and cost also give to each schedule.

- AutoSend_Request/AutoSend_Notify: request/response to send requests to a service, broker will do the automatic scheduling to some severs to achieve total minimum response time.

- ScheSend_Request/ScheSend_Notify: request/response to send requests based on a certain scheduling.

- ScheToServer_Request/ScheToServer_Notify: request/response to connect directly to some servers based on user selection.

In general, this substrates provide mechanisms to tailor the transparent control to satisfy different requirements of the CSCW application.

5.2.2 Dynamic Scheduling

A dynamic scheduling mechanism is implemented in our system. It is used for broker to match servers in case more than one server is available. Since the requests are routed by a broker to servers, the results are passed back to the client through the broker, the dynamic scheduling scheme will not introduce many overhead. And because a broker works as a server to other brokers, our dynamic scheduling method balances loads not only among servers, but also among brokers.

5.2.3 Common Task Sharing

Common task sharing performed by brokers is an important optimization task implemented in our system. The key issue here is how brokers can detect a common task, since the meaning of a common task varies from task to task. The distribution substrate provides a mechanism to let a user indicate his/her preference meaning of a common task. Currently, there are several system predefined meanings of a common task which can be selected by users. As a last solution, the user can describe the detecting functions and hand them to the system. Broker will detect the common task by user's functions.

This substrate only provides the support and control mechanisms, it leaves the policy decision to applications or users.

6 Applications

There are the many possible user, broker and task configuration scenarios in the Shastra brokered collaborative system.

6.1 Cooperative Design

An example of a multi-user cooperative design in Shastra is Collaborative Smoothing using Shilp and Ganith toolkits [2]. This application permits a group of collaborating Shilp users to collectively smooth out a rough polyhedral model by fitting C^1 continuous patches using Hermite interpolation [5]. The Ganith Algebraic Geometry Toolkit is optimized to perform algebraic manipulation – curve-curve, curve-surface, and surface-surface intersection, as well as interpolation. The Shilp Geometric Design and Modeling Toolkit is optimized for boundary representation based solid modeling. Generation of the surface patch is a compute intensive operation. The actual interpolation operation is performed by using instances of the Ganith Toolkit, or Ganith servers, Figure 4.

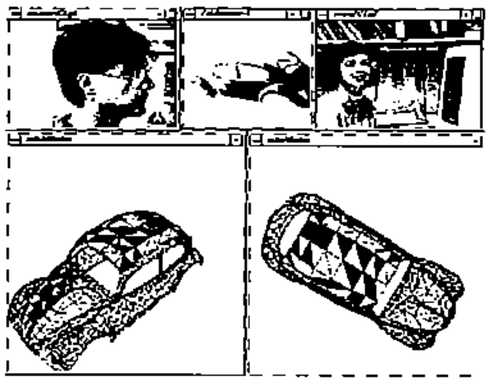
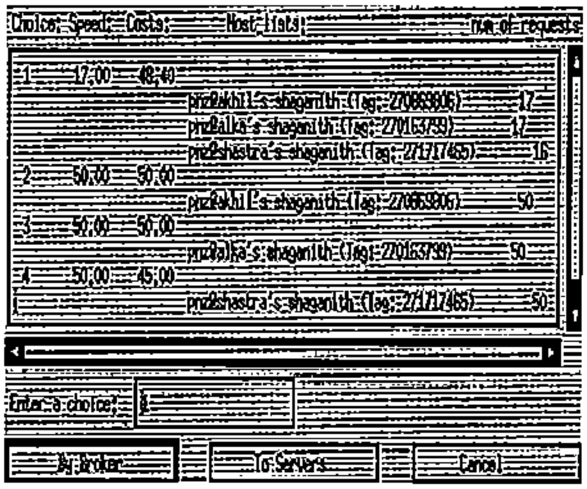


FIGURE 4: Collaborative Smoothing using Shilp and Ganith toolkits with broker support

In the brokered setting, the Shilp instances communicate with their broker. The broker based on the load information of each machine creates multiple Ganith server instances on idle machines on the network, and return the speeds and costs information to the Shilp instances. Shilp sends multiple patch computation requests to a broker. On one way, the broker transparently passes the request to Ganith servers according to the dynamic, adaptive load balance policy which is sensitive to the changes of load on the server machines. On the other way, the broker can generate a set of possible schedules based on current machines' load information, Shilp can select a schedule according to its time and cost consideration. After the selection, the requests are then serviced on the connected servers in keeping with the schedule. See Figure 5. This setup significantly improves the throughput of large design tasks.



```
Choice: Speed: Costs:        Host lists:                         num of requests
1     17.00   48.40
                    pnz@akhil's shaganith (Tag: 270869806)          17
                    pnz@alka's shaganith (Tag: 270163799)           17
                    pnz@shastra's shaganith (Tag: 271717465)        16
2     50.00   50.60
                    pnz@akhil's shaganith (Tag: 270869806)          50
3     50.00   50.00
                    pnz@alka's shaganith (Tag: 270163799)           50
4     50.00   45.00
                    pnz@shastra's shaganith (Tag: 271717465)        50

Enter a choice: 1
```

FIGURE 5: A set of possible schedules for Shilp in Collaborative Smoothing

6.2 Volume Visualization

Volume visualization is a very intuitive method for interpretation of volumetric data [11]. It provides mechanisms to express information contained in typically huge, data sets via images. The synchronously conferenced collaborative volume visualization environment in Shastra [4] lets multiple users on a network share volume data sets, simultaneously view shaded volume renderings of the data, and interact with multiple views. It supports several ways of viewing volumetric data. Facilities are provided for interactive control and specification of the visualization process.

Visualizing volumes is data computationally intensive. Large data sets are visualized using brokers which partitions image space (the volumetric data set) appropriately and use a pool of visualization servers on the network to generate the final image. The brokers use load balancing and scheduling strategies to optimize total rendering time. The most important feature is common task sharing mechanism that let multiusers share common images or parts of images to avoid needless recomputation. As Figure 6 shows, two people collaboratively visualize the same data set with different cutaways. Broker allows them to share the same data part to avoid duplicated computing.

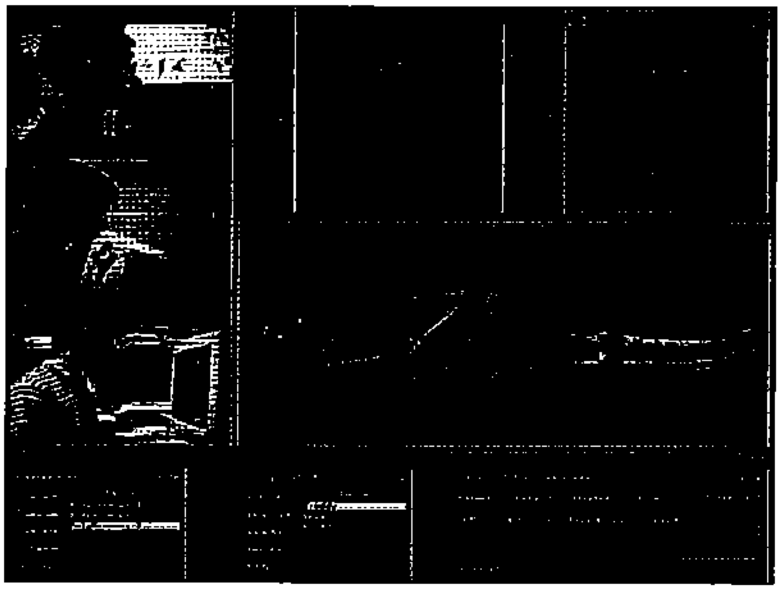


FIGURE 6: Distributed collaborative visualization with broker support

7 Features and Future Work

In this paper, we introduced the concept of Object Request Broker for CSCW systems. We developed a brokered, collaborative infrastructure underlying the current well developed distributed techniques. This infrastructure provides an efficient collaboration support for a hierarchy of sessions, allowing collaboration among groups as well as within a group. It also improves throughput of the system by balancing system load and exploiting the plurality and commonality of tasks in a cooperative setting. The brokered infrastructure makes CSCW system easy to integrate and extend.

We need to explore formal information representation mechanisms to describe tasks to brokers.

References

[1] S. Ahuja, J. Enson, and D. Horn. The rapport multimedia conferencing system. In *Proceedings of ACM Conference on Office Information Systems*, 1988.

[2] V. Anupam and C. Bajaj. Collaborative Multimedia Scientific Design in SHASTRA. In *Proc. of the First ACM International Conference on Multimedia, ACM MULTIMEDIA 93*, pages 447–456. ACM Press, 1993.

[3] V. Anupam and C. Bajaj. SHASTRA - An Architecture for Development of Collaborative Applications. In *Proc. of the Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 155–166. IEEE Computer Society Press, 1993.

[4] V. Anupam, C. Bajaj, D. Schikore, and M. Schikore. *Distributed and Collaborative Volume Visualization.* Computer Science Technical Report, CAPO-93-50, Purdue University, 1993.

[5] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics*, 19(1):61–91, January 1992.

[6] Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick, and Raymond Tomlinson. Mmconf: An infrastructure for building shared multimedia applications. In *CSCW' 90*, pages 329 – 342, 1990.

[7] G. A. Ellis, Simon J. Gibbs, and Gail L. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1), January 1991.

[8] T.Todd Elvins and David R. Nadeau. Netv: An experimantal network-based volume visulization system. In *1991 IEEE Visualization*, pages 239 – 245, 1991.

[9] T.C Nicholas Graham and Tore Urnes. Relational view as a model for automatic distributed implemenation of multi-user applications. In *CSCW' 92*, pages 59–66, 1992.

[10] R. Hill. Languages for construction of multi-user, multi-media synchronous (mumms) applications. 1992.

[11] A. Kaufman. *Volume Visualization*. IEEE Computer Society Press Tutorial, New York, 1990.

[12] W. H. Kohler. A survey of techniques for synchronisation and recovery in decentralsied computer systems. *ACM Computer Surveys*, 13(2), June 1981.

[13] OMG. The common object request broker: Architecture and specification. Technical report, OMG, 1991.

[14] John F. Patterson, Ralph D. Hill, and Steven L. Rohall. Rendezvous: An architecture for synchronous multi-user applications. In *CSCW' 90*, pages 317 – 328, 1990.

[15] Richard L. Phillips. Distributed visualization at los alamos national laboratory. *Computers*, 22(8):70–77, August 1989.

[16] M. Roseman and S. Greenberg. A groupware toolkit for building real-time conferencing applications. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, 10 1992.

[17] A. S. Tanenbaum and R. V. Renesse. Distributed operating systems. *ACM Computer Surveys*, 17(4):419–470, Dec. 1985.